CSE 158/258, DSC 256, MGTA 461, Fall 2023: Homework 1

Instructions

Please submit your solution **by Monday Oct 16.** Submissions should be made on **gradescope**. Please complete homework **individually**.

You should submit two files:

`answers_hw1.txt` should contain a python dictionary containing your answers to each question. Its format should be like the following:

```
{ "Q1": 1.5, "Q2": [3,5,17,8], "Q2": "b", (etc.) }
```

The provided code stub demonstrates how to prepare your answers and includes an answer template for each question.

`homework1.py` A python file containing working code for your solutions. The autograder *will not execute your code*; this file is required so that we can assign partial grades in the event of incorrect solutions, check for plagiarism, etc. Your solution should **clearly document which sections correspond to each question and answer**. We may occasionally run code to confirm that your outputs match submitted answers, so **please ensure that your code generates the submitted answers.**

You will need the following files:

Homework 1 stub : `https://cseweb.ucsd.edu/classes/fa23/cse258-a/stubs/`

GoodReads Fantasy Reviews :
`https://cseweb.ucsd.edu/classes/fa23/cse258-a/data/fantasy_10000.json.gz`

Beer Reviews : `https://cseweb.ucsd.edu/classes/fa23/cse258-a/data/beer_50000.json`

The above are *json* formatted datasets. Code to read them is included in the stub.

Further code examples for regression and classification are available on the class and textbook webpages. Executing the code requires a working install of Python 2.7 or Python 3 with the scipy packages installed.

Each question is worth one mark unless otherwise specified.

Tasks — Regression (week 1):

First, using the *book review* data, let's see whether ratings can be predicted as a function of review length, or by using temporal features associated with a review.

1. Train a simple predictor that estimates rating from review length, i.e.,

$$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{review length in characters}].$$

   Rather than using the review length directly, scale the feature to be between 0 and 1 by dividing by the maximum review length in the dataset. Report the values θ_0 and θ_1, and the Mean Squared Error of your predictor (on the entire dataset).

2. Extend your model to include (in addition to the scaled length) features based on the time of the review. You can parse the time data as follows:

```
> import dateutil.parser
> t = dateutil.parser.parse(d['date_added'])
> t.weekday(), t.month # etc.
```

Using a *one-hot encoding* for the weekday and month, write down feature vectors for the first two examples. Be careful not to include any redundant dimensions: e.g. your feature vector, including the offset term and the length feature, should contain no more than 19 dimensions.

3. Train models that

   - use the weekday and month values directly as features, i.e.,

     $$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{review length in characters}] + \theta_2 \times [\text{t.weekday()}] + \theta_3 \times [\text{t.month}]$$

   - use the one-hot encoding from Question 2.

   Report the MSE of each.

4. Repeat the above question, but this time split the data into a training and test set. You should split the data into 50%/50% train/test fractions **following the split used by the code stub**. After training on the training set, report the MSE of the two models (the one-hot encoding from Question 2 and the direct encoding from Question 3) on the test set.

Tasks — Classification (week 2):

In this question, using the *beer review* data, we'll try to predict ratings (positive or negative) based on characteristics of beer reviews. Load the 50,000 beer review dataset, and construct a label vector by considering whether a review score is four or above, i.e.,

```
y = [d['review/overall'] >= 4 for d in dataset]
```

5. Fit a logistic regressor that estimates the binarized score from review length, i.e.,

   $$p(\text{rating is positive}) \simeq \sigma(\theta_0 + \theta_1 \times [\text{length}])$$

   Using the `class_weight='balanced'` option, report the number of True Positives, True Negatives, False Positives, False Negatives, and the Balanced Error Rate of the classifier.

6. Plot the precision@K of your classifier for $K \in \{1, 100, 1000, 10000\}$.

7. Improve your predictor (specifically, reduce the balanced error rate) by incorporating additional features from the data (e.g. beer styles, ratings, features from text, etc.). Describe your improvement (as a string) and report the BER of your new predictor (2 marks).